JarvisOJ RE 部分writeup

原创

43v3rY0unG 于 2020-01-03 15:19:46 发布 100 收藏
分类专栏： # RE 文章标签： CTF RE

本文链接：https://blog.csdn.net/weixin_43876357/article/details/103816252

0x00 FindKey

这题还蛮有意思的，拿到文件之后，有一个小坑，就是要判断它是什么类型的文件，需要怎么转换。拿到ubuntu里面，file一下查看文件类型（这个方法是从大佬的博客上看到的：

转载自：https://blog.csdn.net/xiangshangbashaonian/article/details/82598211），然后推测是pyc反编译。

（这里插入一个关于pyc的知识：pyc 是由py文件经过编译后二进制文件，py文件变成pyc文件后，加载的速度有所提高，而且pyc是一种跨平台的字节码，是由python 的虚拟机来执行的。）

然后用cmd命令提示符反编译该文件，得到一段python代码

```
C:\Users\hp>uncompyle6 F:\JarvisOJ\findkey.pyc
# uncompyle6 version 3.3.5
# Python bytecode 2.7 (62211)
# Decompiled from: Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC
# Embedded file name: findkey
# Compiled at: 2016-04-30 17:54:18
import sys
lookup = [
 196,
 153, 149,
 206, 17,
 221, 10, 217, 167, 18, 36, 135, 103, 61, 111, 31, 92, 152, 21, 228, 105, 191, 173, 41, 2, 245,
, 182, 119, 38, 85, 48, 226, 165, 241, 166, 214, 71, 90, 151, 3, 109, 169, 150, 224, 69, 156, 1
 51, 252, 227, 93, 65, 82, 66, 80, 170, 77, 49, 177, 81, 94, 202, 107, 25, 73, 148, 98, 129, 23
 171, 64, 180, 233, 74, 140, 242, 75, 104, 253, 44, 39, 87, 86, 27, 68, 22, 55, 76, 35, 248, 96
8, 220, 72, 100, 247, 8, 63, 249, 145, 243, 155, 222, 122, 32, 43, 186, 0, 102, 216, 126, 15, 4
9, 204, 117, 223, 141, 159, 131, 232, 124, 254, 60, 116, 46, 113, 79, 16, 128, 6, 251, 40, 205,
9, 184, 201, 110, 255, 26, 91, 211, 132, 160, 168, 154, 185, 183, 244, 78, 33, 123, 28, 59, 12,
209, 108, 235, 237, 118, 101, 24, 234, 106, 143, 88, 9, 136, 95, 30, 193, 176, 225, 198, 197, 1
1, 70, 58, 187, 50, 67, 236, 230, 13, 99, 190, 208, 207, 7, 53, 219, 203, 62, 114, 127, 125, 16
0, 133, 130, 52, 189, 97, 146, 34, 157, 120, 195, 45, 4, 142, 139]
pwda = [188, 155, 11, 58, 251, 208, 204, 202, 150, 120, 206, 237, 114, 92, 126, 6, 42]
pwdb = [53, 222, 230, 35, 67, 248, 226, 216, 17, 209, 32, 2, 181, 200, 171, 60, 108]
flag = raw_input('Input your Key:').strip()
if len(flag) != 17:
```


然后简单修改一下，写个脚本，得到flag：

```
import sys
lookup = [
 196,
 153, 149,
 206, 17,
 221, 10, 217, 167, 18, 36, 135, 103, 61, 111, 31, 92, 152, 21, 228, 105, 191, 173,
pwda = [188, 155, 11, 58, 251, 208, 204, 202, 150, 120, 206, 237, 114, 92, 126, 6, 4
pwdb = [53, 222, 230, 35, 67, 248, 226, 216, 17, 209, 32, 2, 181, 200, 171, 60, 108]

flag = ""
for i in range(0, 17):
    flag += chr(lookup[(i + pwdb[i])] - pwda[i] & 255)
flag = flag[::-1]
print(flag)
```



```
D:\PyCharm\PycharmProje
PCTF{PyC_Cr4ck3r}
```

0x01 Classical CrackMe

这种题第一次做，放进ida里什么都没...于是借鉴了一下大佬的博客，拖进PEID发现是用C#编写的，于是拖进.NET Reflector查看，这个软件我也是第一次用...所以借鉴了一下大佬的博客，才算摸透一些。但其实这个题本身的思路不难，就是一个base64解密的过程。这里我就不写出来了，可以借鉴一下大佬的wp，转载链接：https://blog.csdn.net/xiangshangbashaonian/article/details/82561920

网上搜了一些关于reflector的资料：Reflector支持四种语言：IL，VB.net，C#，Delphi；而IDA是可以把程序反编译成c语言的软件（怪不得放进ida里那么奇怪.....）

0x02 Smali

这道题还学到了蛮多新知识的，首先拿到一个smali文件，搜一下有关smail文件的知识，指路链接：https://blog.csdn.net/lixpjita39/article/details/75193833

然后用smalijava2UI这个软件吧smail文件转换为Java文件，（其实硬看也可以看懂大致的意思）指路链接：https://blog.csdn.net/SHENGLI_509/article/details/73520571

看了一下，str2用base64解密当作key，然后AES128解密得到flag，学着写了一个python写了一个脚本：（第一次写AES解密的脚本）

```python
from Crypto.Cipher import AES

import base64

key='cGhyYWNrICBjdGYgMjAxNg=='
cipher='sSNnx1UKbYrA1+MOrdtDTA=='

str1 = base64.b64decode(key)
str2 = base64.b64decode(cipher)

aes = AES.new(str1,AES.MODE_ECB)
print(aes.decrypt(str2))
```


期间还搜了一下pycharm怎么导入crypto包，还学到了蛮多知识吧，最后flag：PCTF{Sm4liRiver}